

(19)日本国特許庁 (J P)

(12) 公表特許公報 (A)

(11)特許出願公表番号

特表2001-508908

(P2001-508908A)

(43)公表日 平成13年7月3日(2001.7.3)

(51)Int.Cl.

G 0 6 F 9/30

識別記号

3 1 0

F I

G 0 6 F 9/30

テ-マコード* (参考)

3 1 0 E

審査請求 未請求 予備審査請求 未請求(全 49 頁)

(21)出願番号 特願平11-521337
 (86)(22)出願日 平成10年9月21日(1998.9.21)
 (85)翻訳文提出日 平成11年6月2日(1999.6.2)
 (86)国際出願番号 P C T / I B 9 8 / 0 1 4 5 3
 (87)国際公開番号 W O 9 9 / 1 8 4 8 6
 (87)国際公開日 平成11年4月15日(1999.4.15)
 (31)優先権主張番号 9 7 2 0 3 0 3 3 . 2
 (32)優先日 平成9年10月2日(1997.10.2)
 (33)優先権主張国 ヨーロッパ特許庁 (E P)
 (31)優先権主張番号 9 7 2 0 3 9 0 5 . 1
 (32)優先日 平成9年12月12日(1997.12.12)
 (33)優先権主張国 ヨーロッパ特許庁 (E P)

(71)出願人 コーニンクレッカ フィリップス エレクトロニクス エヌ ヴィ
 オランダ国 5621 ベーアー アイन्दーフェン フルーネヴァウツウェッハ 1
 (72)発明者 オーフステイン アレクサンデル
 オランダ国 5658 アーアー アイन्दーフェン プロフ ホルストラーン 6
 (72)発明者 デュクストラ エールコ イェー
 オランダ国 5656 アーアー アイन्दーフェン プロフ ホルストラーン 6
 (74)代理人 弁理士 杉村 暁秀 (外2名)

最終頁に続く

(54)【発明の名称】 可変命令セットコンピュータ

(67)【要約】

ソースプログラムが、処理ユニット(100)のマイクロコントローラコア(114)上で実行される。該コア(114)は、マイクロコントローラ固有の命令の所定セットからのネイティブ命令を実行することができる。前処理ステップにおいて、上記ソースプログラムのプログラム命令文に対して、プログラム固有の仮想マシンが対応する仮想マシン命令のセットと共に次のように定義される。即ち、上記プログラム命令文の命令系列での表現が、ネイティブ命令のみを使用した場合に比べて少ない記憶空間しか必要としないように定義される。上記プログラム固有の仮想マシンに対して、該プログラム固有の仮想マシンの命令を上記コア(114)のネイティブ命令に変換するための関連する変換手段(132)が定義される。上記ソースプログラムの命令文は定義された仮想マシンの命令を含む命令の系列で表現される。この命令の系列は命令メモリ(120)に記憶される。変換手段(132)は処理ユニット(100)内で表される。実行の間に、命令メモリ(120)から命令が取り込まれる。そして、変換手段(132)は、上記の取り込まれた仮想マシン命

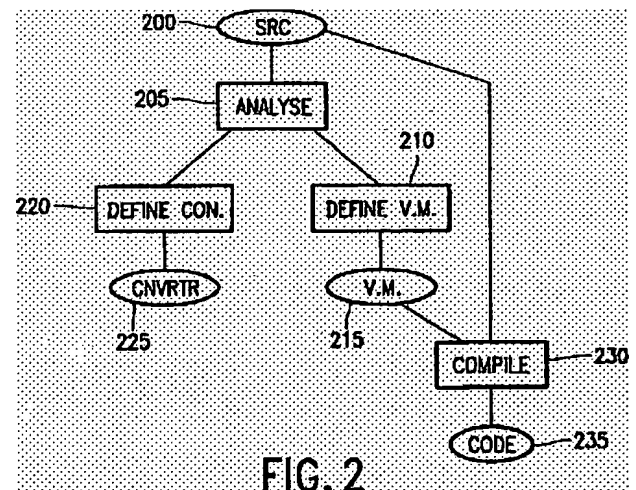


FIG. 2

【特許請求の範囲】

1. ソースプログラムを、所定のマイクロコントローラ固有の命令のセットからのネイティブ命令を実行する所定のマイクロコントローラコアを有するような処理ユニット上で実行する方法であって、該方法が、

- ー 前記ソースプログラムのプログラム命令文を仮想マシン命令を有する命令の系列で表現すると共に、該命令の系列を命令メモリに記憶する前処理ステップと、
- ー 前記命令メモリから命令を取り込むと共に、該命令メモリから取り込まれた仮想命令を前記処理ユニットの変換手段を用いてネイティブ命令に変換し、且つ、これらネイティブ命令を実行のために前記マイクロコントローラコアに供給するような実行ステップと、

を有するような方法において、前記前処理ステップが、

前記ソースプログラムのプログラム命令文に対して、対応する仮想マシン命令のセットを伴うプログラム固有の仮想マシンを、前記プログラム命令文の前記命令の系列での表現が該プログラム命令文を表現するのにネイティブ命令のみを使用した場合に較べて前記命令メモリ内に少ない記憶空間しか必要としないような形態で、定義し、

前記プログラム固有の仮想マシンに対して、該プログラム固有の仮想マシンの仮想マシン命令を前記マイクロコントローラコアのネイティブ命令に変換するような関連する変換手段を定義し、

前記関連する変換手段を前記処理ユニット内で表す、
ような各ステップを有していることを特徴とするソースプログラムを処理ユニット上で実行する方法。

2. 請求項1に記載のソースプログラムを処理ユニット上で実行する方法において、前記前処理ステップが、

- ー 前記ソースプログラム内のプログラム命令文の複数の群を識別し、

- ー これらプログラム命令文の群の各々に対して、

プログラム群固有の仮想マシンを対応する仮想マシン命令のセットと共

に定義し、

前記プログラム群固有の仮想マシンに対して、仮想マシン命令を前記マイクロコントローラコアのネイティブ命令に変換するような関連する変換手段を発生し、

前記関連する変換手段を前記処理ユニット内で表し、

前記処理ユニット内に、発生された各仮想マシン命令を前記群固有の変換手段に関連付ける選択データを記憶する、

ような各ステップを有し、前記実行ステップが、取り込まれた命令に対して前記選択データにより示される前記関連する変換手段を選択するステップを有していることを特徴とする方法。

3. 請求項1に記載の方法において、該方法が、

他の仮想マシンの仮想マシン命令を含む命令を有するような命令モジュールを入力すると共に、前記処理ユニットの他の変換手段が前記他の仮想マシン命令をネイティブ命令に変換することができるようにする変換データを入力し、

前記命令モジュールを前記命令メモリに記憶し、

前記変換データを前記処理ユニットに記憶し、

前記他の仮想マシン命令の各々を前記変換データに関連付ける選択データを前記処理ユニットに記憶する、

ような各ステップを有し、前記実行ステップが、

取り込まれた他の仮想マシン命令に対して、前記選択データにより示される前記関連する変換データを選択し、

前記他の変換手段を前記選択された変換データの制御の下で動作させる、

ような各ステップを有していることを特徴とする方法。

4. 仮想マシンの仮想マシン命令と呼ぶ命令を実行する処理ユニットであって、

所定のマイクロコントローラ固有の命令のセットからの、前記仮想マシン命

令とは異なるネイティブ命令を実行する所定のマイクロコントローラコアと、

前記仮想マシン命令の少なくとも1つを含む命令を記憶する命令メモリと、

前記命令メモリから取り込まれた仮想マシン命令を、前記マイクロコントロ

ーラコアにより実行するための少なくとも1つのネイティブ命令に変換する変換手段を有するコンバータと、

を有するような処理ユニットにおいて、

前記コンバータが複数の異なる仮想マシンに対して前記変換を実行するように動作することを特徴とする処理ユニット。

5. 請求項4に記載の処理ユニットにおいて、前記変換手段が再プログラム可能な形式のものであることを特徴とする処理ユニット。

6. 請求項4に記載の処理ユニットにおいて、前記コンバータが前記複数の仮想マシンの各々に対して、対応する仮想マシンの仮想マシン命令を変換するための変換手段を有していることを特徴とする処理ユニット。

7. 請求項6に記載の処理ユニットにおいて、該処理ユニットが前記命令メモリ内の少なくとも2つの分離された群のロケーションを各変換手段に関連付ける選択データを有し、前記コンバータが前記命令メモリ内のロケーションから取り込まれた命令を前記選択データに基づいて前記変換手段に選択的に指向させることを特徴とする処理ユニット。

8. 請求項4に記載の処理ユニットにおいて、該処理ユニットは、前記ネイティブマシンと前記仮想マシンとの間及び／又は異なる仮想マシンの間の区別をするために、命令に対してメモリから関連する選択データを取り込むように動作し、前記コンバータが前記取り込まれた命令を前記選択データに基づいて変換手段に選択的に指向させる検出器を有していることを特徴とする処理ユニット。

9. 請求項4に記載の処理ユニットにおいて、前記コンバータは1つの仮想マシン命令を1つの対応するネイティブ命令に変換するように動作し、前記仮想マシン命令は対応するネイティブ命令よりも前記命令メモリ内に要する記憶空間に関して一層コンパクトにコード化されていることを特徴とする処理ユニット。

10. 請求項4に記載の処理ユニットにおいて、前記コンバータは1つの仮想マシン命令を複数のネイティブ命令の所定の系列に変換するように動作し、該処理

ユニットが、

前記コンバータと前記マイクロコントローラコアとの間に結合されて、前記ネイティブ命令の系列を前記マイクロコントローラコアに順次供給するシーケンサと、

上記供給の間に、前記命令メモリからの命令の取り込みを禁止する禁止手段と、

を有することを特徴とする処理ユニット。

11. 請求項10に記載の処理ユニットにおいて、前記禁止手段が前記禁止を前記マイクロコントローラコアの命令ポインタのインクリメントを妨害することにより実行するよう動作することを特徴とする処理ユニット。
12. 請求項11に記載の処理ユニットにおいて、該処理ユニットは、当該命令取り込み部の命令カウンタの変化に応答して前記命令メモリから命令を取り込む命令取り込み部を有し、前記命令カウンタは前記マイクロコントローラコアの命令ポインタの変化に応答して異なる値に設定され、前記禁止手段は前記命令カウンタの値の変化を妨害することにより前記禁止を実行するよう動作することを特徴とする処理ユニット。

【発明の詳細な説明】

可変命令セットコンピュータ

本発明は、マイクロコントローラ固有の命令の所定のセットからのネイティブ命令を実行するための所定のマイクロコントローラコアを有するような処理ユニット上でソースプログラムを実行する方法に係り、該方法が、

ー 上記ソースプログラムのプログラム命令文を仮想マシン命令（仮想機械語命令）を有する命令の系列で表現すると共に該命令の系列を命令メモリに記憶する前処理ステップと、

ー 上記命令メモリから命令を取り込むと共に、該命令メモリから取り込まれた仮想命令を前記処理ユニットの変換手段を用いてネイティブ命令に変換し、更に、これらネイティブ命令を実行のために前記マイクロコントローラコアに供給するような実行ステップと、

を有するような方法に関する。

本発明は、更に、仮想マシンの仮想マシン命令と呼ばれる命令を実行する処理ユニットであって、

マイクロコントローラ固有の命令の所定のセットからの上記仮想マシン命令とは異なるネイティブ命令を実行する所定のマイクロコントローラコアと、

上記仮想マシン命令の少なくとも1つを含む命令を記憶する命令メモリと、

上記命令メモリから取り込まれた仮想マシン命令を、上記マイクロコントローラコアにより実行するための少なくとも1つのネイティブ命令に変換するための変換手段を有するコンバータと、

を有するような処理ユニットにも関する。

益々、ソースプログラムは、該プログラムが実行されるべきマイクロコントローラコアのネイティブ命令の代わりに、仮想マシンの命令で表現される（コンパイルされる）ようになってきている。仮想マシンを用いる主たる理由は、異なる

マシン（プラットフォーム）の間でのプログラムの移植性である。仮想マシンの仮想マシン命令で表現されたプログラムは、幾つかの具体的マシン上で動作する適切なインタプリタを用いることにより、これらマシン上で比較的容易に実行す

ることができる。現時点では、移植性のあるプログラムを使用しようとする推進力はジャバであり、ジャバプログラムはインターネットを介して交換されると共に、異なる命令セットを持つプロセッサを用いた異なるネイティブマシン上で実行することができる。コンパイラを用いて、ジャバプログラムは、ジャバ仮想マシンの命令を形成するジャババイトコード(JBC)で表現される。結果としてのコードは、通常、ジャバアプレットと称される。

従来、仮想マシン命令で表されるプログラムはソフトウェア翻訳により実行されている。プロセッサ(CPU)が特別なインタプリタプログラムを実行し、そこでは、ループ内で該プロセッサが仮想マシン命令を取り込み、該命令を当該プロセッサのマイクロコントローラコアのネイティブ命令の系列にデコードし、そして各ネイティブ命令を実行する。この技術は、遅く、しかも比較的大きな付加的なインタプリタプログラムを必要とする。実行速度を改善するため、所謂ジャスト・イン・タイム(JST)コンパイル技術が使用される。仮想マシン命令で表されたソフトウェアモジュールの実行を開始する直前に、該モジュールはネイティブコードにコンパイルされる(即ち、ネイティブマシン命令で表される)。このようにして、該モジュールは当該コンパイラ用のコードに加えて、2度記憶されねばならない。ソフトウェア翻訳の上記の付加的な記憶要件は、組み込み型のシステムにとっては望ましいものではない。性能及び記憶上のオーバーヘッドを避けるために、ハードウェアインタプリタを使用することが好ましい。ハードウェアインタプリタそれ自体は、ウォレンの抽象命令セット用のプロログ(Prolog)プリプロセッサ(pre-processor: 前処理プログラム)の形で知られている。1986年のマイクロプロセッシング及びマイクロプログラミングの第71~81頁における、B・クネドラー及びW・ローゼンステールによる“ウォレンの抽象命令セット用のプロログプリプロセッサ”なる論文には、プロログプロ

グラミング言語で書かれたプログラムをモトローラの68000プロセッサ(MC68000)上で翻訳するためのプリプロセッサが記載されている。該プロログソースプログラムを命令に翻訳するためにコンパイラが使用され、これら命令はウォレン氏により定義されたもので、通常、プロログプログラムを実行するために使用され

る。これらのウォレン命令のセットは、プロログプログラムを実行するために設計された仮想マシンを形成する。上記コンパイルの結果としてのウォレン命令の系列はRAMにロードされ、上記プリプロセッサの助けでMC68000により実行される。電源投入後、上記MC68000はネイティブなMC68000命令を実行することにより、先ずブート処理を行う。該ブート処理の後、該MC68000はプロログ命令の実行を開始することが可能となる。これは、所定のアドレス範囲にジャンプすることにより開始される。上記プリプロセッサはメモリにマップされる装置であり、該装置は上記範囲にマップされている。このプリプロセッサがアドレス指定されると、該プリプロセッサは自身のRAMから（翻訳されたプロログプログラムの）ウォレン命令を読み出し、MC68000命令の系列及び定数を適応的に合成し、これらを実行のために直接CPUに送出する。各ウォレン命令に対するMC68000命令は上記プリプロセッサのROMに記憶されている。通常、該プリプロセッサは1つのウォレン命令を1つの系列のMC68000命令に翻訳する。該プリプロセッサは自身のRAMコントローラ及びROMコントローラを含み、これらコントローラが当該プリプロセッサのRAM及びROM用のアドレスを発生する。RAMコントローラはRAM命令のポインタを管理する。MC68000の順次の読み出し動作の各々の結果、プリプロセッサは上記系列の次の命令（及び随意的定数）をCPUに送ることになる。当該系列が完了すると、次の読み取り動作の結果として、当該プログラムの次のウォレン命令に対応する系列の最初の命令がCPUに送出される。この既知のプリプロセッサは1つの仮想マシン（ウォレンマシン）をサポートする。

本発明の一つの目的は、もっと柔軟性のある上述したような種類の方法及び処理ユニットを提供することにある。また、本発明の他の目的は、プログラムがも

っとコンパクトな形で表される上述したような種類の方法及び処理ユニットを提供することにある。

本発明の上記目的を達成するため、本方法は、

前記ソースプログラムのプログラム命令文に対して、対応する仮想マシン命令のセットを伴うプログラム固有の仮想マシンを、上記プログラム命令文の前記命

令系列での表現が該プログラム命令文を表現するのにネイティブ命令のみを使用した場合に較べて前記命令メモリ内に少ない記憶空間しか必要としないような形態で定義し、

上記プログラム固有の仮想マシンに対して、該プログラム固有の仮想マシンの仮想マシン命令を前記マイクロコントローラコアのネイティブ命令に変換するような関連する変換手段を定義し、

該関連する変換手段を前記処理ユニット内で表す、
ような各ステップを有していることを特徴としている。

本発明によれば、プログラムに対してプログラム固有の仮想マシンが、該プログラムが前記コアのネイティブ命令で表現された場合よりも該プログラムが一層コンパクトな形で表現することができるように、定義される。また、関連する変換手段も定義される。この変換手段は、例えば、ROMに記憶された変換テーブル若しくはマイクロコード又はPLDのような専用のロジックを使用して実施化することができる。定義された変換手段は、前記処理ユニット内で表される。このようにして、プログラム的高速実行が維持される一方、同時にコンパクトな表現が達成される。この方法は、組み込み型アプリケーションと共に使用するのに特に好適である。この場合、上記ソースプログラムは、当該組み込み型システム内で最初に表される全てのプログラム命令文に関係する（例えば、ユーザがシステムを購入した時に当該システム内に存在するプログラム）。該プログラムはROMのような永久メモリに記憶することもでき、又はEEPROMのような再プログラム可能なメモリに記憶することもできる。組み込み型アプリケーションの場合は、該組み込み型アプリケーションのプログラムを表すのに使用されるコードがコンパクト

であり、該コードを実行する性能が良好であることが高度に望まれる。典型的には、組み込み型アプリケーション用に用いられる処理ユニットはファミリー思想に基づくものであり、そこでは、特定のアプリケーションに対して、処理ユニットが、該アプリケーションに必要とされる所与のマイクロコントローラコア及びI/O又は記憶部品から作成される。経費削減のため、上記コア及び部品

に対しては何の又は少なくとも大きな変更はしないことが望ましい。本発明によれば、一方では、当該プログラムに対して特別に仮想マシン命令が定義されて、コードのコンパクト化を達成するための十分な柔軟性を与え、他方では、該プログラムを実行するために在庫品的な（即座に入手可能な）コアが使用される。仮想マシン命令からネイティブ命令への所要な変換は変換手段により実行され、該手段は一種のプリプロセッサと見ることができる。

1995年1月のPOPL'95の第322～332頁におけるトッド・A・プロエブスティングによる“ANSI Cインタプリタのスーパー演算子を用いた最適化”なる文献は、バイトコードによるインタプリタ用の最適化技術としてのスーパー演算子の使用について記載している。最初のステップとして、プログラムを仮想マシン命令で表すためにコンパイラが使用される。この仮想マシン命令は、該仮想マシン命令を用いたプログラムの表現が一層コンパクトになるように、選択される。このような仮想マシンの一例はスタックマシンであり、斯かるスタックマシンは、そのコンパクトな表現で知られている。次に、コンパイルされたコード内で頻繁に発生する仮想マシン命令の系列が識別され、新たに定義された追加の仮想マシン命令により置換されるが、その場合、例えば1つの新たな命令（所謂、スーパー演算子）が4つの既存の命令の系列を置換する。スーパー演算子は、コンパイラの構文解析相の間に、当該コンパイラの間接表現の間に構築される表現ツリーにおいて最も頻繁に発生する項目を識別することにより定義することもできる。限られた数（例えば、256を大きく下回る）のスーパー演算子のみが定義される。これにより、スーパー演算子のコンパクトなコード化（例えば、単一バイトのコードを用いる）が可能となる。この場合、該コードの範囲の

一部が元の仮想マシン命令を表すために割り当てられ、該範囲の一部は上記スーパー演算子を表す新たに追加した仮想マシン命令を表すために使用される。20個のみのスーパー演算子を用いることにより、既に大幅なコードのコンパクト化が達成される。これら仮想マシン命令は、MIPS R3000又はSPARCプロセッサ上で走るソフトウェアインタプリタを用いて翻訳される。ハードウェアイン

タブリタは使用されず、これは所与の組み込み型マイクロコントローラコアとの組み合わせで、性能上の不利益なしに所与の組み込み型アプリケーションプログラムのコードのコンパクト化をもたらす。

請求項2に記載した実施例においては、異なる群のプログラム命令文に対して、良好なレベルのコードのコンパクト化を達成することができる。好ましくは、ネイティブ命令の同一の副セットに実質的に関係するプログラム命令文は一緒に群にまとめられ、斯かる群に対して仮想マシンが定義される。例えば、浮動小数点の演算に対するよりも、整数の演算に対しては異なる仮想マシンが定義される。ネイティブ命令の略同一の副セットに関係するが、これら命令をかなり相違する頻度で使用するプログラム命令文は群にまとめるのが好ましく、これにより最も頻繁に発生する命令（の系列）に対しては特別な仮想マシン命令の定義を可能にする。マルチタスクシステムの場合は、好ましくは、1つのタスクのプログラム命令文は1つの仮想マシンに関連付けられる。これは、組み込み型のプログラムの開発を簡単にする。何故なら、通常は1つのタスクに関するプログラム命令文は一人の技術者の管理の下で作成され、このやり方においては、上記管理の役割をコンパクト化されたコード及び関連する変換手段の制作の管理を含むように拡張することができるからである。選択された構成に依存して、実行されるべき仮想マシン命令に関連付けられた変換手段の選択が、実行の遅延を招き得る。この場合、個々の仮想マシン命令に対して変換手段を選択する代わりに、変換手段をタスクの切り換えの一部として切り換えることによりオーバーヘッドを低減することができる。この場合、同一のタスクを切れ目無く実行する限り、適切な変換手段を選択するためのオーバーヘッドが関わることはない。同一の仮想マシンは、

ネイティブ命令の同様の混合を同様の頻度で使用する幾つかのタスクに対しても使用することができることが分かるであろう。一方、タスク用のプログラムが2以上のプログラムモジュールを有し、その場合に、これらプログラムモジュールの少なくとも2つが、ネイティブ命令の異なる副セットに実質的に関係するか、又は同様の命令の副セットの命令を実質的に異なる頻度で使用するような場合は

、これらのモジュールに対しては異なる仮想マシンを使用するのが好ましい。同様に、オブジェクト指向のプログラミング技術が用いられる場合は、仮想マシンは同様のオブジェクトの各々又は群に対して定義することができる。

請求項3に記載の実施例においては、命令モジュールと、該モジュールの命令をネイティブ命令に変換するための変換データとが入力される。例えば、当該処理ユニットには追加の又は置換用のプログラムモジュールが供給される。該供給は、例えばプログラムモジュールを備えるROMを追加する又は交換することによりローカルなものであってもよく、又は該プログラムモジュールをフロッピーディスクのような可搬型記憶媒体からプログラム可能型メモリにロードすることでもよい。上記供給は、ローカルエリアネットワーク又はインターネットのような広域ネットワークの何れかのネットワークを介してなされてもよい。益々、組み込み型のシステムは、追加のソフトウェア、又は既存のソフトウェアの一部を当該システムの初期プログラミングの後で置換するようなソフトウェアを受け付けるように“開放型(open)”になってきている。上記初期ソフトウェアが専用の仮想マシンを用いて最適に圧縮されていた場合は、該マシンを新たに入力されたコードに対して使用することは、常に良好な結果を与えるとは限らず、不可能な場合さえある。元のソフトウェアの供給者が新たなソフトウェアを供給する場合でさえも、該ソフトウェアの性質が異なる場合があり、これは、該新たなソフトウェアに対しては別の仮想マシンを定義することにより一層良好なコンパクト化が達成される可能性があることを意味する。益々、入力されるソフトウェアが、元のソフトウェアとは完全に異なる性質のものであるようになってきている。一例として、新たなソフトウェアはジャバのバイトコードで表現されたジャバアプリ

レットであるかもしれず、一方、元のソフトウェアは“C”で書かれており、元のプログラムに適合した仮想マシンと対応する変換手段とに最適にコンパイルされている。この定義された変換手段はジャババイトコードを変換するために使用することはできないことが理解されるであろう。何故なら、これらのコードは異なる仮想マシンを用いて表現されているからである。

本発明の前記目的を達成するため、本処理ユニットは、前記コンバータが複数の異なる仮想マシンに対して変換を実行するように動作することを特徴としている。

例えば、組み込み型のシステムには最初に或る仮想マシンを用いて表された組み込み型のプログラムが供給され、好ましくは、この仮想マシンは該特定のプログラムに対して定義される。後の段階で、当該ソフトウェアの略全部又は一部を取り替えるか又はソフトウェアモジュールを追加するようなソフトウェアの更新が必要となる。その場合に形成される全体のアプリケーションプログラムに対しては、その時点で有効なプログラム（依然として古いプログラムの部分も含むことができる）を一層良好に反映するような新たな仮想マシンを使用するのが望ましい。特に、ソフトウェアの所要の量は時間と共に増加する傾向にあるから、新たに生成されるアプリケーションプログラムに対しては一層高いレベルのコードのコンパクト化（異なる仮想マシンに反映されて）が必要であろう。このようにして、時間につれて異なる仮想マシンが使用され、その場合に、各時点では唯一の仮想マシンが使用される。他の例として、前述したように、異なる仮想マシンが組み込み型アプリケーションプログラムの異なる部分に対して同時に使用されるようにしてもよい。異なる仮想マシンは、“C”で書かれ且つプログラム固有の仮想マシンを用いてコンパクト化された組み込み型プログラム及び異なり且つ通常は予め決められている仮想マシンを用いて表現されたジャバアプレット等のプログラムのような、出所の異なるプログラムに対して使用することもできる。複数の仮想マシンは当該処理ユニット内に同時に存在することもでき、その場合には、時間につれて新たな仮想マシンが追加されるか又は既存の仮想マシンが置

換されることが分かるであろう。

請求項5に記載の処理ユニットの実施例においては、前記変換手段は再プログラム可能な形式のものである。これによれば、新たなプログラムがロードされる場合に、新たな変換手段を当該処理ユニットに“ダウンロード”することができるようになる。上記の再プログラム可能な変換手段は、例えば、(E)EPROM等に記憶された再プログラム可能な変換テーブル若しくはマイクロコードを用いて、又

はE-P L Dのような専用の再プログラム可能なロジックを用いて、実施化することができる。

請求項6に記載の処理ユニットの実施例においては、前記コンバータは仮想マシンの各々に対して専用の変換手段を有している。この実施例では、処理ユニットには幾つかの専用の変換手段が作成される。原理的に、1つの対応する変換手段を備える1つの大きな仮想マシンを使用することもできる。幾つかの仮想マシンを使用することにより、一層コンパクトなコードを達成することができる。例えば、組み込み型のプログラムが2つのプログラムモジュールにより形成されていると仮定して、これらのモジュールのうち一方のモジュールは当該組み込み型システムのユーザインターフェースとしての面に主に関係し、かくして主にネイティブ整数命令を必要とし、第2モジュールは主に信号処理に関係して、主に浮動小数点命令を必要とする。更に、上記2つのモジュールの各々に対して最適の仮想マシンは256の仮想マシン命令を各々有し、これらモジュールが8ビットのコードを用いて表現されると仮定する。両モジュールをカバーする1つの仮想マシンは上記2つの仮想マシンの命令を合成することにより作成することができる。512までの（少なくとも256を越える）仮想マシン命令を得ることができる。結果として、該コードの大きさは増加する。何故なら、この場合は9ビットのコードが必要となるからである。このような増加は、通常、可能性のある命令の重なり合いにより、合成された変換テーブルは2つの別個の変換テーブルより小さいという事実によっては補償することはできない。

請求項7に記載の処理ユニットの実施例においては、前記コンバータは異なる

仮想マシンに対する変換手段の間を、実行されるべき仮想命令が記憶される命令メモリ内のロケーションに基づいて、区別する。例えば、ネイティブ命令とは別に2つの異なる仮想マシンの仮想マシン命令も使用される場合は、該メモリは3つの領域、即ち各形式の命令に対して1つの領域、に分割することができる。実行されるべき該命令のアドレスが何の領域に存在するかを判定することにより、上記コンバータは、変換が必要（ネイティブ命令が変換されるべき）か否か、及び何の変換手段が該変換に使用されるべきかを容易に検出することができる。

請求項8に記載の処理ユニットの実施例においては、ネイティブマシンと仮想マシンとの間及び／又は異なる仮想マシンの間を区別するために、別個の指示子（例えば、1以上のビット）が使用される。別個の指示子は各命令に対して使用することができる。他の例として、マシンの変更が生じる毎に設定されるような1つの包括的な指示子（例えば、レジスタ）を使用することもできる。

請求項9に記載の処理ユニットの実施例においては、1つの仮想マシン命令が一層コンパクトにコード化される1つの対応するネイティブ命令に変換される。このようにして、該変換は非常に単純となる。

請求項10に記載の処理ユニットの実施例においては、1つの仮想マシン命令は複数のネイティブ命令の所定の系列に変換され、更なるレベルのコンパクト化を提供する。ネイティブ命令の該系列の前記マイクロコントローラコアへの供給を、例えば該系列が供給されている間は上記マイクロコントローラの命令ポインタ（プログラムカウンタ）のインクリメントを禁止すると共に、一連動作が完了した場合にインクリメントを可能にすることにより制御するために、シーケンサが使用される。他の例として、当該処理ユニットは、自身の命令カウンタの制御の下で前記命令メモリから命令を取り込む命令取り込み部を有する。一連動作が完了すると、該カウンタの変化が可能にされ、マイクロコントローラコアの命令ポインタ（プログラムカウンタ）の変化に応答して、該命令カウンタは異なる値に設定される。系列が処理されている間は、上記命令カウンタの値の変化は禁止される。

本発明の上記及び他の特徴は図を参照して説明される実施例から明らかになるであろう。

図1は、処理ユニット内にコンバータを配置する4つの可能性のある構造的選択例を示し、

図2は、仮想マシン及び関連するコンバータを規定する処理を示し、

図3は、プログラム命令文の幾つかのコヒーレントな群から形成されるプログラムに関する処理を示し、

図4は、コンバータのブロック図を示し、

図1は、処理ユニット100内にコンバータを配置する4つの可能性のある構造的選択例を図示している。該処理ユニット100の3つの主要部分は、マイクロコントローラ110、命令メモリ120及びプリプロセッサ130である。プリプロセッサ130はコンバータ132を有する。示される全ての図において、マイクロコントローラ110は命令メモリ120とプリプロセッサ130とを有している。処理ユニット100それ自体は明示的には示されていない。マイクロコントローラ110内の全ての主たる要素を組み合わせることにより（好ましくはワンチップ装置とする）、最適な性能が達成される。もし所望なら、命令メモリ120及び／又はプリプロセッサ130はマイクロコントローラ110の外に配置することもでき、その場合は、マイクロコントローラバス140が該マイクロコントローラ110の外まで延長され、例えば、PCIのような外部バスに結合されるようにすることもできることが分かる。

命令メモリ120は、スタックマシン用の命令のような仮想マシン命令を含んでいる。該命令メモリはデータを記憶するためにも使用することができる。本発明は、データ及び命令が別れているハーバードアーキテクチャに限定されるものではない。マイクロコントローラ110は、マイクロコントローラに固有の命令の所定のセットからのネイティブ命令を実行するための、ネイティブマシンと呼ばれる、所定のマイクロコントローラコア114を備えるようなプロセッサ112を有している。組み込み型のソフトウェアを実行するのに適したマイクロコン

トローラの一例は、MIPS PR3001範囲のマイクロプロセッサのようなRISC型のマイクロコントローラである。上記プロセッサは、ネイティブ命令を、これらの命令を実行する前に記憶するための命令キャッシュ116を有することができる。該マイクロコントローラコア114のネイティブ命令は、前記仮想マシンの仮想マシン命令とは異なる。マイクロコントローラ110それ自体は、命令メモリ120に記憶された仮想マシン命令を直接実行することはできない。命令を要求するプロセッサ112に応答して、プリプロセッサ130がネイティブ命令を出力する。該ネイティブ命令を発生することができるように、プリプロセッサ130は、取り込み手段134を用いて、命令メモリ120から仮想マシン命令を取

り込むことができる。プリプロセッサ130のコンバータ132は、命令メモリ120から取り込まれた仮想マシン命令を少なくとも1つのネイティブ命令に変換するために使用される。通常、仮想マシン命令はネイティブ命令の系列に変換される。プリプロセッサ130は、更に、上記系列のネイティブ命令を実行のためにマイクロコントローラコア114に供給する供給手段136を有している。仮想マシンプログラムを実行する場合、マイクロコントローラ110は、実際に、プリプロセッサ130により発生されたネイティブプログラムを実行する。マイクロコントローラ110の命令ポインタが通常命令メモリ120内の当該マイクロプロセッサ110により次に実行される必要のある次の命令を示す場合、ここでは、該命令ポインタはプリプロセッサ130に対して次のネイティブ命令（又は前の命令の再供給）が必要であることを示す。結果として、プリプロセッサ130は、命令メモリ120内の現在の（又は次の）仮想マシン命令を示す独立した仮想マシン命令ポインタを管理することになる。当該マイクロコントローラは、仮想マシン命令又は仮想マシン命令ポインタを（明確に）知る必要はない。

図1Aにおいて、当該処理ユニットの主要な部分は、PIバスのような汎用周辺装置相互接続バス140を介して相互接続されている。プリプロセッサ130は該バス上の周辺装置である。プリプロセッサ130はメモリにマップされた周辺装置として動作し、そこでは、該プリプロセッサに対し所定の範囲のアドレス

が割り当てられている。プロセッサ112がバス140上に上記範囲内のアドレスを伴う命令の要求を送出することに応答して、プリプロセッサ130はバス140上にネイティブ命令を出力する。必要があれば、プリプロセッサ130はバス140を介して命令メモリ120から仮想マシン命令を取り込む。

図1B及び図1Cにおいては、プリプロセッサ130はプロセッサ112と命令メモリ120との間に位置されている。プリプロセッサ130がネイティブ命令と仮想マシン命令とを区別する必要がある場合は、これらの構成は、命令メモリ120に記憶されたネイティブ命令の実行を遅延させることができる。明瞭化の理由で、図1Aに示された全ての要素が図1B、図1C及び図1Dで繰り返されていない。

図1Dにおいては、プリプロセッサ130はプロセッサ112内に組み込まれている。プリプロセッサ130は、好ましくは、プロセッサ112の命令キャッシュ116とコア114との間に位置される。この構成は最適な性能を可能にするが、図1A、1B及び1Cの構成とは違って、マイクロコントローラ110の変更を必要とし、プリプロセッサ130それ自体は、同一形式のコア114を備える異なる形式のプロセッサに対しては、在庫設計のようには、使用することはできない。

前処理ステップにおいては、当該処理ユニット100により実行されるべきソースプログラムのプログラム命令文に対して、対応する仮想マシン命令のセットを伴うプログラム固有の仮想マシンが定義される。更に、上記のプログラム固有の仮想マシンに対して、対応するコンバータ132が仮想マシン命令をマイクロコントローラコアのネイティブ命令に変換するために定義される。該コンバータ132は、上記仮想マシン命令の各々に対して、ネイティブコア114の、該仮想マシン命令を実行するために必要とされる1個の又は通常は一連の対応するネイティブ命令を供給する。図2は上記仮想マシン及び関連するコンバータを定義する過程を図示している。ステップ205においては、ソースプログラム200が解析される。この解析（例えば、演算の発生頻度）に基づいて、ステップ21

0ではプログラム固有の仮想マシン215が定義される。ステップ220においては、該仮想マシン用に、関連するコンバータ225が定義され、ステップ230においては上記ソースプログラムが該仮想マシンの命令で表され、結果としてコード235が得られる。上記仮想マシンは、前記プログラム命令文を表すのにネイティブ命令のみを使用するのに較べて、上記コードが命令メモリ内に必要とする記憶空間が少なくなるように、定義される。

好ましくは、上記のプログラム固有の仮想マシンは、上記プログラムを所定の仮想マシンの仮想マシン命令に変換することにより定義される。開始点として、コンパクトな表現が得られるものとして知られているスタックマシンのような仮想マシンが選択される。スタックマシンはオペランドを記憶するために明確なレジスタは使用しないが、その代わりに、オペランドはスタックに記憶され、演算

子は常に該スタックの最上要素（又は複数の要素）に作用するという事実により、スタックマシンの場合は、使用することが可能なオペランドの数はあまり限定されず、これらオペランドの管理が一層簡単である。これにより、スタックに基づくマシンはレジスタに基づくマシンの場合よりも、コンパイラを構築するのが容易となる。更に、スタックマシンの命令は、殆どのレジスタに基づくマシンよりも単純な構造を持つ傾向がある。しかしながら、好適なレジスタに基づくマシンを使用することもできることが分かるであろう。本発明を説明するため、付録Aのスタックマシンが使用される。“C”プログラム言語で書かれたサンプルプログラム（8人の女王の問題を解く）が付録Bに示されている。該プログラムの付録Aの仮想マシンのスタックコードへの変換が付録Cに示されている。該コードを解析することにより、頻繁に発生する命令の系列が識別され、各々がこれらの識別された系列の1つを表すような追加の仮想マシン命令が定義される。付録Dは、好適な追加のマシン命令のリストを示している。このような追加のマシン命令は、“スーパー仮想マシン命令”と見ることができる。次いで、上記プログラム固有の仮想マシンが、付録Aの基本命令と付録Dのスーパー命令との組み合わせで形成される。付録Eは、付録Bのプログラムの付録A及びDの仮想マシンへ

の表現を示している。次に、上記のプログラム固有の仮想マシンに対して、図1に示すような関連するコンバータ132が、仮想マシン命令を前記マイクロコントローラコアのネイティブ命令に変換するために定義される。該コンバータ132は、上記仮想マシン命令（本例では、付録A及びDの命令の組み合わせ）の各々に対して、該仮想マシン命令をネイティブコア114上で実行するために要する1つの又は通常は一つの系列の対応するネイティブ命令を供給する。このような方法を用いれば、ネイティブ命令を表すのに通常16又は32ビットが使用されるのと較べて、各パラメータ化されていない命令に対して短い表現（例えば、7ビット）を可能にするような比較的少数の仮想マシン命令（本例では、38個の基本命令及び40個のスーパー命令）を用いることにより、上記コードのコンパクト化が達成される。上記スーパー命令を使用する結果、当該プログラムを表

すのに少ない命令しか必要とされなくもなる。本例では、当該プログラムを基本仮想マシンの命令で表すのには356の命令文が必要である。スーパー命令を使用すると、これが262の命令文だけが必要となるように低減される。これは、当該プログラムがネイティブ命令で表現され、そこではMIPSプロセッサに対しては608個の命令文が必要となるような状況に較べると有利である。

好ましくは、プログラムに固有な仮想マシンは以下のようにして定義される。ソースプログラムは、所定の仮想マシンに基づいて所謂表現ツリーの中間レベル表現に変換される。表現ツリーは、該仮想マシンの基本演算を表すノードにより構成される。該表現ツリーは、該ツリーのノードの基本演算を所定の順序（例えば、接尾辞順）で実行する結果としての計算を表す。当該プログラムの表現ツリーへの変換は、フレーザ及びハンソンの「ライト“C”コンパイラlcc」のような、コンパイラの構文解析相を用いて実行される。次に、該表現ツリーとコヒーレントなツリー断片（一連の演算を表す）とが識別される。これらツリー及び断片の各々は、スーパー命令により表されるべき候補である。前記基本仮想マシン命令から始めて、最もコードサイズの節約が得られるようなツリー及びツリー断片が、スーパー命令により表される。該スーパー命令は前記仮想マシンに追加さ

れる。当該プログラムは、節約を決定するために、該新たな仮想マシン命令のセットにより表される。新たなスーパー命令は、新たな命令を追加する余裕が依然としてある限り（例えば、仮想マシン命令がバイトとして記憶される必要がある場合は、命令の数は256に限定される）、又は最早節約が達成されなくなるまで、追加される。最後に、当該ソースプログラムは上記プログラム固有の仮想マシンの命令で表され、変換データが発生される。

当該プログラムが、一旦、仮想マシン命令で表されると、結果としてのコードは命令メモリ120に記憶される。組み込み型システムの場合は、該コードは通常ROMに記憶されるが、該ROMはマスクプログラムされたROM、又はPROM若しくは(E)EPROMのようなプログラム可能なROMとする。同様に、上記の発生された変換データは処理ユニット100のコンバータ132内で表される。該コンバータ132は、例えば、ROMに記憶される変換テーブル又はマイ

クロコードを用いて実施化することができる。また、コンバータ132は、PLDのようなロジックを用いて実施化することもできる。再プログラム可能なコンバータの場合は、同様の方法を、(E)EPROM又はE-PLDのような各々の再プログラム可能な技術に基づいて使用することができる。

図3は本発明による他の実施例を示し、該図において、例えば“C”でプログラムされたソースプログラム300はステップ310において、プログラムモジュール、オブジェクト又はタスク固有のコードのような幾つかのコヒーレントなプログラム命令文の群に分割される。図示のものは、モジュール312及び316である。明確な分割は必要ではなく、ソースプログラムが既に好適なモジュール構造で利用可能であるかもしれないことが分かるであろう。群312及び316の各々に対して、図2のステップ205及び210と同様の方法により、プログラム群固有の仮想マシンが、対応する仮想マシン命令のセットと共に定義される(ステップ330)。図示したものは、各仮想マシン332及び336である。ステップ340においては、各群312及び316のプログラム命令文が各仮想マシン332及び336の命令で表され、結果として各コードモジュール342

及び346が得られる。これらコードモジュールはステップ350において命令メモリ120に記憶される。ステップ360においては、上記プログラム群固有の仮想マシン332及び336に対して、仮想マシン命令をマイクロコントローラコアのネイティブ命令に変換するための変換手段362及び366が各々発生される。ステップ370においては、これら変換手段362及び366が当該処理ユニット100内で、例えば該処理ユニット100内に変換テーブル又は変換ロジックをプログラムすることにより表される。結果として、コンバータ132は上記プログラム群の各々に対して固有の変換手段を有することになる。処理ユニット100が取り込まれた仮想命令に関連する群固有の変換手段を選択することができるように、ステップ390においては、選択データが当該処理ユニットに記憶される。実行の間には、仮想マシン命令が命令メモリ120から取り込まれる。この場合、上記選択データが当該仮想マシン命令が属する仮想マシンに関

連する変換手段を捜すために使用される。

図4はコンバータ132のブロック図を示し、該図においてコンバータ132は幾つかの変換手段(400、410及び420が図示されている)を有している。これら変換手段の各々が、固有の仮想マシンの仮想マシン命令を変換するために使用される。該コンバータ132は、各々が完全な変換を実行することができるような完全に自律的な変換手段を有してもよいことが分かるであろう。他の例として、コンバータ132は、別個の変換データ(テーブルのような)の制御の下で仮想マシンの各々に対して何らかの共有ロジックを使用することもできる。当該処理ユニット100は選択データ430を有する。該選択データは、命令メモリ120に記憶される、又は当該変換手段が(部分的に)データとして記憶される場合は該変換手段と組み合わせて記憶される等の如何なる好適な方法でも表すことができる。また、選択データ430はメモリからロードされるコンバータ132内のレジスタの形を採ることもできる。種々の形の選択データを使用することができる。

他の実施例においては、命令メモリ120のアドレス範囲が幾つかの副範囲に分割される。一つのアドレス副範囲は1つの仮想マシンのみの仮想マシン命令を記憶するために確保される。好ましくは、一つのアドレス副範囲はネイティブ命令を記憶するためにも確保される。このようなネイティブ命令は、例えば、当該システムを初期化するか、又はドライバ若しくは組み込み型ソフトウェアアプリケーションの特別な部分のようなソフトウェアモジュールが最適な性能のためにネイティブ命令にコンパイルされるのを可能にするために使用される。この実施例では、前記選択データは、規定されたアドレス副範囲の各々に関して、命令取り込み部134により取り込まれた命令をネイティブ命令に変換するために上記変換手段400、410及び420のうちの何れが使用されるべきかを示す。この目的のため、コンバータ132は、命令メモリ120内のロケーションから取り込まれた命令を上記選択データに基づいて変換手段400、410又は420のうちの1つに選択的に指向させる検出器440を有している。ネイティブ命令も命令メモリ120内に記憶されている場合は、該検出器440は、これら命令

がマイクロコントローラコア114へ供給されるように供給器136に直接供給されるのを保証する。

上記判断が取り込まれた命令のアドレスに基づく代わりの例として、情報が該命令に直接関連付けられて記憶されるようにすることもできる。例えば、命令メモリ120への各エントリの1以上のビットが、仮想マシン間の及び／又はネイティブコードと仮想コードとの間の区別をつけるために確保されるようにすることもできる。例えば、命令が7ビットを要するような場合に2つの仮想マシンのみが使用されるとすると、8番目のビットは当該命令が属する仮想マシンと関連する変換手段とを示すために使用することができる。明らかに、このような技術はアドレスに基づく区別と組み合わせることもできる。

他の例として、上記選択データはレジスタに記憶されてもよく、その場合、該レジスタは仮想マシンの間の切り換えが発生する毎に、別の変換手段を示すように設定される。該レジスタを設定するためには、特別な命令（例えば、ジャンプ命令の一形態）を使用することができる。斯かる構成をサポートする命令は、命

令メモリ内の仮想マシン命令／ネイティブ命令と混ぜることができる。

上述したように、典型的には、仮想マシン命令はネイティブ命令の系列に変換される。命令の前記コアへの供給を調節するために、当該処理ユニット100は、コンバータ132とマイクロコントローラコア114との間に結合されて上記ネイティブ命令の系列をマイクロコントローラコア114に順次供給するシーケンサ450を有する。該シーケンサ450はカウンタのような通常の部品を用いて実施化することが可能であり、該カウンタは、例えば、当該マイクロコントローラコアからの新たな命令が必要であることを示すトリガ（例えば、当該コアの命令カウンタのインクリメント）の結果としてインクリメントされる。通常の処理ユニットにおいては、マイクロコントローラコアの命令ポインタ（プログラムカウンタとも称される）の変化の結果として、取り込み部134は命令メモリ120から命令を取り込み、供給器136は該命令をマイクロコントローラコア114に供給する。自動的な取り込みと供給との間の結合を絶つため、当該処理ユニット100は、更に、前記系列の供給の間は命令メモリからの命令の取り込み

を禁止するための禁止手段460を有している。

上記順次供給及び禁止は幾つかの方法で実施することができるであろう。他の実施例においては、禁止手段460は上記禁止をマイーラコア114の命令ポインタのインクリメントを妨害することによつて動作する。これには、コア114に対する小さな変更を必要と、コア114への追加の制御ラインにより命令カウンタの選択的禁り、その場合、上記禁止手段460は前記系列からの命令が供給されるインクリメントを禁止し、系列が完全に供給されたらインクリメントする。

他の実施例においては、命令取り込み部134が、マイクロコン114の命令ポインタとは別に、自身の命令カウンタを維持する。0は、何時マイクロコントローラの命令ポインタのインクリメント的には変化)の結果、命令取り込み部134の命令カウンタ135を制御する。上述したのと同様に、禁止手段460は前記系列から

されている場合は命令カウンタ135の変化を禁止し、系列が完全ら変化を可能にする。命令カウンタ135の変化の結果、命令取りは、通常、命令メモリ112の該命令カウンタにより示されるアドレスな命令を取り込む。有利には、仮想マシン命令のコンパクトな表現、命令取り込み部134はメモリから数個の命令を1つの読み出しむことができる(例えば、4つの1バイト命令を1つの32ビット読み出すことができる)。このように、命令カウンタ135の全ても結果として新たな命令の実際の取り込みになる必要はない。

本発明による他の実施例においては、他の仮想マシンの仮想マシンれたプログラムモジュールが、例えば、ネットワークを介して又はクラウドメモリから入力される。このようなプログラムモジュール、ジャバアプレットである。入力されたプログラムモジュールは命令0に記憶される。該他の仮想マシン命令のネイティブ命令への変換めるため、変換データも入力される。該変換データは、例えば、テ

く変換手段用の変換テーブル、又はPLDに基づく変換手段用のE-PLDプログラミングデータを規定する。該変換データは、コンバータ132の他の変換手段による後の使用のために、当該処理ユニットに記憶される。実行の間に適切な変換手段を選択することができるように、他の各仮想命令を上記変換データに関連付ける選択データも当該処理ユニットに記憶される。実行の間には、取り込まれた他の仮想マシン命令に関して、上記他の変換手段は上記選択データにより示される変換データの制御の下で動作される。

上述したように、コンバータ132は仮想マシン命令をネイティブ命令の系列に変換するためのテーブルを有することができる。一次元テーブルを使用することができ、その場合、該テーブルの各セルは、対応する1つの仮想マシン命令に対して一連のネイティブ命令を有する。セル番号が、対応する仮想マシン命令の値に対応してもよい。一例として、ジャバの整数加算(0x60)用のネイティブ命令の系列は、セル96(16進で0x60に等しい)内に配置することがで

きる。ネイティブ命令の系列の長さは種々の仮想命令に対して大幅に変化するもので、好ましくは、斯かる系列は明確なセル無しに一次元テーブル内に配置され、その場合、各系列が互いに直後に続くようにする。このような変換テーブル500が図5に示されており、該図では暗黙のセル境界が破線を用いて示されている。ある仮想マシン命令に対する系列を捜すことができるように、コード索引テーブル510を使用することができ、該索引テーブルは各仮想マシン命令(VMI1ないしVMIN)に対して、当該変換テーブル500内の対応する系列の開始点を示す。変換テーブル500におけるVMI3に対応するセルに関して、ネイティブ命令NI1ないしNIMからなる関連する系列520が図示されている。

変換に関する他の例がジャバのバイトコードbipush n(バイトnを符号拡張し、結果をスタックの最上部に配置するために使用される)に関して示される。この仮想マシン命令は2つのバイト{0x16及びn}からなり、ここで、最初のバイトは演算を定義し、第2バイトはパラメータnを提供する。該命令はネイティブMIPS命令の下記の系列に変換することができる。


```

ori $a0, $0, n/* Load register $a0 with constant n */
sll $a0, $a0, 24/* Shift left by 24 bits */
sra $a0, $a0, 24/* Arithmetic shift right, causing sign extension,
by
/* replicating last left-most bit */
sw $a0, 0 ($tosp)/* Store result at new top of stack */
addi $tosp, -4/* Increment stack size */

```

この例は、仮想マシン命令がパラメータ化され、そこでは、演算コードには少なくとも1つのオペランドが続くことを示している。有利には、コンバータ132は変換テーブル500を有し、そこでは、ネイティブ命令は完全なコードにより、又は命令スケルトンにより、のいずれかで表される。一例として、上記命令addi \$tosp, -4（上記例の系列の最後の命令）は変数部分を有さず、したがって当該テーブル内に4バイトエントリとして全て配置することができる。上記命令ori

\$a0, \$0, n（上記例の系列の最初の命令）は変数部分を含んでおり、したがって該変数部分（nである）を特定せずに当該テーブル内にスケルトンとして配置される。好ましくは、命令スケルトン用の当該テーブルへのエントリは完全命令（例えば、MIPSプロセッサの場合は4バイト）と同じ幅とし、一様なテーブルを可能とする。該テーブル（又は他の（複数の）テーブル）には、どの様にしてネイティブ命令スケルトンの特定されていない部分を埋めるべきかを示す他の情報を配置することができる。有利には、上記の特定されていない部分を埋めるために、マイクロプログラミングが使用される。この場合、上記の他の情報はマイクロコードを有するか又はマイクロコードを示す。命令スケルトンに対しては完全ネイティブ命令と同じ構成（幅及び組立）を使用するのが有利であることが分かるであろう。しかしながら、他の構成も同様に使用することができる。

当該仮想マシンがスタック指向のマシンである場合は、好ましくは、スタック又は該スタックの少なくとも最上側の要素はマイクロコントローラ110のレジスタ上にマップされるようにする。このようにして、メモリストック（仮想マシ

ンスタックを伴う) はレジスタスタックにマップされる。レジスタ\$r1、\$r2及び\$r3が当該メモリスタックの3つの連続した要素を含んでおり、その場合に、最初は\$r1が当該メモリスタックの第1の空のロケーション(該スタックの最上部の上)に対応し、\$r2が当該メモリスタックの最上部を含み、\$r3が当該メモリスタックの第2の要素を含んでいると仮定すると、ジャバのバイトコードbipush nはネイティブMIPS命令の下記系列に変換することができる。

```
ori $r1, $0, n  
sll $r1, $r1, 24  
sra $r1, $r1, 24
```

この演算の後、\$r1は当該メモリスタックの最上部を含むようになる。

同様にして、整数加算(0x60)用のジャババイトコード(仮想マシン命令)は、最初は\$r1が当該メモリスタックの第1の空のロケーション(該スタックの最上部の上)に対応し、\$r2が当該メモリスタックの最上部を含み、\$r3が当該メモリス

タックの第2の要素を含んでいるような同一の位置から始めて、MIPS命令の下記系列に変換することができる。

```
add $r3, $r2, $r3
```

この演算の後、\$r3は当該メモリスタックの最上部を含むようになる。

上記例において、好ましくは、メモリスタックの最上部の位置(即ち、どのレジスタが当該メモリスタックの最上部を含むか)は、コンバータ132のレジスタ138を用いて示されるようにする。該コンバータは、レジスタスタックポインタ(RSP)と呼ばれるレジスタ138を、適切なネイティブ命令を発生するために使用する。好ましくは、ネイティブ命令のレジスタオペランドを特定するためにマイクロプログラミングが使用される。このようにして、固定のネイティブ命令も可変となる。何故なら、レジスタオペランドはコンバータ132により特定される必要があるからである。好ましくは、このようなオペランドも命令スケルトンを用いて変換テーブル500に記憶されるようにする。RSPが最初の

空きのレジスタを指すと仮定すると、ジャババイトコードhipush nは、対応するマイクロコードの制御の下でネイティブMIPS命令の下記の系列に変換することができる。

マイクロコード命令

```

rsp = 1;          ftg = rsp + lori $(rsp + 1), $0, n
ftg = rsp + 1; fa0 = rsp + 1; lsl1 $(rsp + 1), $(rsp + 1), 24
ftg = rsp + 1; fa0 = rsp + 1; lsr1 $(rsp + 1), $(rsp + 1), 2

```

ここで、f_{tg}は当該命令用の目標レジスタを示し、f_{a0}及びf_{a1}は当該命令用の第1及び第2引数レジスタを各々示す。スタックの最上側の2つの要素を加算するための、続くジャババイトコードiaddは、結果として、下記のマイクロコード及び命令となる。

```

ftg = rsp + 2; fa0 = rsp + 2; fa1 = rsp + 1; rsp += iadd $(rsp + 2), $(rsp + 2), $(rsp + 1)

```

付録A、基本仮想マシン命令

演算子	形式接尾辞	演算
ADDRF	P	パラメータのアドレス
ADDRG	P	グローバルのアドレス
ADDRL	F	ローカルのアドレス
CNST	CSIUPFD	定数
BCOM	U	ビット単位補数
CVC	IU	文字からの変換
CVD	IF	倍からの変換
CVF	D	浮動からの変換
CVI	CSUD	整数からの変換
CVP	U	ポインタからの変換
CVS	IU	短からの変換
CVU	CSIP	無符号からの変換
INDIR	CSIPFDB	取り込み

NEG	IFD	否定
ADD	IUPFD	加算
BAND	U	ビット単位のアンド
BOR	U	ビット単位のアオ
BXOR	LT	ビット単位の排他的オア
DIV	IUFD	除算
LSH	IU	左シフト
MOD	IU	モジュロ
MUL	IUFD	乗算
RSH	IU	右シフト
SUB	IUPFD	減算
ASGN	CSIPFDB	割り当て
EQ	IFD	等しいならジャンプ
GE	IUFD	大きい又は等しいならジャンプ
GT	IUFD	大きいならジャンプ
LE	IUFD	小さい又は等しいならジャンプ
LT	IUFD	小さいならジャンプ
NE	IUFD	等しくないならジャンプ
ARG	IPFDB	引き数
CALL	IFDBV	関数呼び出し
RET	IFD	関数からの戻り
JUMP	V	無条件ジャンプ
LABEL	V	定義
ADDSP		スタックポインタのインクリメント
POP		スタックをポップ
接尾辞		
B	ブール式	
C	文字	

D	倍
I	整数
F	浮動小数点
P	ポインタ
S	短
U	無符号整数
V	ラベル

付録B. サンプル“C”プログラム


```
int up[15], down[15], rows[8], x[8];
int queens(), print();

main()
{
    int i;
    {
        char *i;
        printf("%s", i);
    }
    for (i = 0; i < 15; i++)
        up[i] = down[i] = 1;
    for (i = 0; i < 8; i++)
        rows[i] = 1;
    queens(0);
    return 0;
}

queens(c)
{
    int r;

    for (r = 0; r < 8; r++)
        if (rows[r] && up[r-c+7] && down[r+c]) {
            rows[r] = up[r-c+7] = down[r+c] = 0;
            x[c] = r;
            if (c == 7)
                print();
            else
                queens(c + 1);
            rows[r] = up[r-c+7] = down[r+c] = 1;
        }
}
```



```
)  
  
print0  
{  
    int k;  
  
    for (k = 0; k < 8; k++) printf ("%c ", x[k] + '1');  
    printf("\n");  
}
```

付録C、デフォルトの仮想マシンで表されたサンプルプログラム

1	.text	30	addp	59	addrp(-4)
2	addrgp(_main)	31	addrp(-12)	60	indri
3	jumpv	32	indri	61	cnsti(1)
4	.globl _main	33	asgni	62	addi
5	_main:	34	L.3:	63	asgni
6	addSP (-12)	35	addrp(-4)	64	addrp(-4)
7	addrp(-4)	36	addrp(-4)	65	indri
8	cnsti(0)	37	indri	66	cnsti(8)
9	asgni	38	cnsti(1)	67	lil(L.6)
10	L.2:	39	addi	68	cnsti(0)
11	addrp(-8)	40	asgni	69	argi
12	addrp(-4)	41	addrp(-4)	70	addrgp(_queens)
13	indri	42	indri	71	calli
14	cnsti(2)	43	cnsti(15)	72	addSP(4)
15	lshi	44	lil(L.2)	73	pop
16	asgni	45	addrp(-4)	74	cnsti(0)
17	addrp(-12)	46	cnsti(0)	75	rer
18	cnsti(1)	47	asgni	76	L.1:
19	asgni	48	L.6:	77	cnsti(99)
20	addrp(-8)	49	addrp(-4)	78	ret
21	indri	50	indri	79	.globl _queens
22	addrgp(_down)	51	cnsti(2)	80	_queens:
23	addp	52	lshi	81	addSP (-52)
24	addrp(-12)	53	addrgp(_rows)	82	addrp(-4)
25	indri	54	addp	83	cnsti(0)
26	asgni	55	cnsti(1)	84	asgni
27	addrp(-8)	56	asgni	85	L.13:
28	indri	57	L.7:	86	addrp(-8)
29	addrgp(_up)	58	addrp(-4)	87	addrp(-4)

88	indiri	119	addrgp(_up+28)	150	addrp(-36)
89	asgni	120	addp	151	cnsti(0)
90	addrp(-12)	121	indiri	152	asgni
91	cnsti(2)	122	addrp(-16)	153	addrp(-24)
92	asgni	123	indiri	154	indiri
93	addrp(-16)	124	eqi(L,17)	155	addrp(-28)
94	cnsti(0)	125	addrp(-8)	156	indiri
95	asgni	126	indiri	157	addi
96	addrp(-8)	127	addrp(-20)	158	addrp(-32)
97	indiri	128	indiri	159	indiri
98	addrp(-12)	129	addi	160	lshi
99	indiri	130	addrp(-12)	161	addrgp(_down)
100	lshi	131	indiri	162	addp
101	addrgp(_rows)	132	lshi	163	addrp(-36)
102	addp	133	addrgp(_down)	164	indiri
103	indiri	134	addp	165	asgni
104	addrp(-16)	135	indiri	166	addrp(-24)
105	indiri	136	addrp(-16)	167	indiri
106	eqi(L,17)	137	indiri	168	addrp(-28)
107	addrp(-20)	138	eqi(L,17)	169	indiri
108	addrfp(8)	139	addrp(-24)	170	subi
109	indiri	140	addrp(-4)	171	addrp(-32)
110	asgni	141	indiri	172	indiri
111	addrp(-8)	142	asgni	173	lshi
112	indiri	143	addrp(-28)	174	addrgp(_up+28)
113	addrp(-20)	144	addrfp(8)	175	addp
114	indiri	145	indiri	176	addrp(-36)
115	subi	146	asgni	177	indiri
116	addrp(-12)	147	addrp(-32)	178	asgni
117	indiri	148	cnsti(2)	179	addrp(-24)
118	lshi	149	asgni	180	indiri

181	addrp(-32)	212	addi	243	addrp(-52)
182	indiri	213	argi	244	indiri
183	lshi	214	addrgp(_queens)	245	asgni
184	addrgp(_rows)	215	calli	246	addrp(-40)
185	addp	216	addSP(4)	247	indiri
186	addrp(-36)	217	pop	248	addrp(-44)
187	indiri	218	L.22:	249	indiri
188	asgni	219	addrp(-40)	250	subi
189	addrfp(8)	220	addrp(-4)	251	addrp(-48)
190	indiri	221	indiri	252	indiri
191	cnsti(2)	222	asgni	253	lshi
192	lshi	223	addrp(-44)	254	addrgp(_up+28)
193	addrgp(_x)	224	addrfp(8)	255	addp
194	addp	225	indiri	256	addrp(-52)
195	addrp(-4)	226	asgni	257	indiri
196	indiri	227	addrp(-48)	258	asgni
197	asgni	228	cnsti(2)	259	addrp(-40)
198	addrfp(8)	229	asgni	260	indiri
199	indiri	230	addrp(-52)	261	addrp(-48)
200	cnsti(7)	231	cnsti(1)	262	indiri
201	nei(L.21)	232	asgni	263	lshi
202	addrgp(_prim)	233	addrp(-40)	264	addrgp(_rows)
203	calli	234	indiri	265	addp
204	addSP(0)	235	addrp(-44)	266	addrp(-53)
205	pop	236	indiri	267	indiri
206	addrgp(L.22)	237	addi	268	asgni
207	jumpv	238	addrp(-48)	269	L.17:
208	L.21:	239	indiri	270	L.14:
209	addrfp(8)	240	lshi	271	addrp(-4)
210	indiri	241	addrgp(_down)	272	addrp(-4)
211	cnsti(1)	242	addp	273	indiri

274	cnsti(1)	305	addSP(8)	336	.space 32
275	addi	306	pop	337	.globl _down
276	asgni	307	L.38:	338	.align 4
277	addrp(-4)	308	addrp(-4)	339	_down:
278	indri	309	addrp(-4)	340	.space 60
279	cnsti(8)	310	indri	341	.globl _up
280	ld(L.13)	311	cnsti(1)	342	.align 4
281	L.12:	312	addi	343	_up:
282	cnsti(99)	313	asgni	344	.space 60
283	ret	314	addrp(-4)	345	.dataEnd
284	.globl _print	315	indri	346	.data
285	_print:	316	cnsti(8)	347	.align 1
286	addSP (-4)	317	ld(L.37)	348	L.42:
287	addrp(-4)	318	addrgp(L.42)	349	.byte 10
288	cnsti(0)	319	argp	350	.byte 0
289	asgni	320	addrgp(_printf)	351	.align 1
290	L.37:	321	calli	352	L.41:
291	addrp(-4)	322	addSP(4)	353	.byte 37
292	indri	323	pop	354	.byte 99
293	cnsti(2)	324	L.36:	355	.byte 32
294	lahi	325	cnsti(99)	356	.byte 0
295	addrgp(_x)	326	ret	357	.dataEnd
296	addp	327	.textEnd		
297	indri	328	.data		
298	cnsti(49)	329	.globl _x		
299	addi	330	.align 4		
300	argi	331	_x:		
301	addrgp(L.41)	332	.space 32		
302	argp	333	.globl _rows		
303	addrgp(_printf)	334	.align 4		
304	calli	335	_rows:		

付録D. スーパー命令


```

1  instr112 : ADDRLP [-4]
2  instr123 : INDIRI(ADDRLP [#])
3  instr298 : EQI [L.17] (INDIRI (ADDP(*, *)), INDIRI(ADDRLP[-16]))
4  instr113 : CNSTI [0]
5  instr133 : CNSTI [1]
6  instr315 : INDIRI (ADDRFP [8])
7  instr124 : CNSTI [2]
8  instr249U : ASGNI (ADDP(LSHI(*, *), ADDRGP[#]), *)
9  instr121 : ADDRLP [-8]
10 instr132 : ADDRLP [-12]
11 instr259U : LTI [#](INDIRI(ADDRLP [-4]), CNSTI[8])
12 instr180 : ASGNI(ADDRLP[-4]ADDI(INDIRI(ADDRLP [-4]), *))
13 instr160 : ASGNI(ADDP(*, ADDRGP[_down]), INDIRI(*))
14 instr355 : ADDRLP [-36]
15 instr471 : ADDRLP [-52]
16 instr243 : ASGNI(ADDP(LSHI(INDIRI(*), *), ADDRGP[_rows]), *)
17 instr352 : ADDRLP [-24]
18 instr468 : ADDRLP [-40]
19 instr354 : ADDRLP [-32]
20 instr470 : ADDRLP [-48]
21 instr514 : ADDI (INDIRI(ADDP(*, ADDRGP[_*])), CNSTI[49])
22 instr211 : LTI [L.2](INDIRI(ADDRLP[-4]), CNSTI[15])
23 instr313 : ADDRLP [-20]
24 instr353 : ADDRLP [-28]
25 instr461 : NEI [L.21](INDIRI(ADDRFP[8]), CNSTI[7])
26 instr469 : ADDRLP [-44]
27 instr114 : ASGNI (ADDRLP[-4], CNSTI [0])
28 instr267 : ADDRGP [_queens]
29 instr309 : ADDRGP [_printf]
30 instr122 : INDIRI ( ADDRLP [-4] )
31 instr336 : ADDRGP [_up+28]
32 instr146 : INDIRI ( ADDRLP [-8] )

```



```
33  instr147 : ADDRGP [_down]
34  instr206 : ADDRGP [_up]
35  instr219 : ADDRGP [_rows]
36  instr276 : ADDRGP [-16]
37  instr473 : ADDRGP [_print]
38  instr474 : ADDRGP [L.22]
39  instr515 : ADDRGP [L.41]
40  instr523 : ADDRGP [L.42]
```

付録E. プログラム固有の仮想マシンで表されるサンプルプログラム

1	.text	31	L.6:	61	asgnl
2	addrpp(_main)	32	instr112	62	instr276
3	jumpv	33	instr124	63	instr113
4	.globl _main	34	instr133	64	asgnl
5	_main:	35	instr243	65	instr146
6	addSP (-12)	36	L.7:	66	instr132
7	instr114	37	instr133	67	indir1
8	L.2:	38	instr180	68	lshl
9	instr121	39	instr259U(L.6)	69	instr219
10	instr122	40	instr113	70	instr298
11	instr124	41	argi	71	instr313
12	lshl	42	instr267	72	instr315
13	asgnl	43	calli	73	asgnl
14	instr132	44	addSP(4)	74	instr146
15	instr133	45	pop	75	instr313
16	asgnl	46	instr113	76	indir1
17	instr146	47	ret	77	subi
18	instr132	48	L.1:	78	instr132
19	instr160	49	cnsti(99)	79	indir1
20	instr146	50	ret	80	lshl
21	instr206	51	.globl _queens	81	instr336
22	addp	52	_queens:	82	instr298
23	instr132	53	addSP (-52)	83	instr146
24	indir1	54	instr114	84	instr313
25	asgnl	55	L.13:	85	indir1
26	L.3:	56	instr121	86	addi
27	instr133	57	instr122	87	instr132
28	instr180	58	asgnl	88	indir1
29	instr211	59	instr132	89	lshl
30	instr114	60	instr124	90	instr147

91	instr298	123	addp	155	instr122
92	instr352	124	instr355	156	asgni
93	instr122	125	indiri	157	instr469
94	asgni	126	asgni	158	instr315
95	instr353	127	instr352	159	asgni
96	instr315	128	instr354	160	instr470
97	asgni	129	indiri	161	instr124
98	instr354	130	instr355	162	asgni
99	instr124	131	indiri	163	instr471
100	asgni	132	instr243	164	instr133
101	instr355	133	instr315	165	asgni
102	instr113	134	instr124	166	instr468
103	asgni	135	instr122	167	indiri
104	instr352	136	instr249U(_x)	168	instr469
105	indiri	137	instr461	169	indiri
106	instr353	138	instr473	170	addi
107	indiri	139	calli	171	instr470
108	addi	140	addSP(0)	172	indiri
109	instr354	141	pop	173	lshi
110	indiri	142	instr474	174	instr471
111	lshi	143	jumpv	175	instr160
112	instr355	144	L.21:	176	instr468
113	instr160	145	instr315	177	indiri
114	instr352	146	instr133	178	instr469
115	indiri	147	addi	179	indiri
116	instr353	148	argi	180	subi
117	indiri	149	instr267	181	instr470
118	subi	150	calli	182	indiri
119	instr354	151	addSP(4)	183	lshi
120	indiri	152	pop	184	instr336
121	lshi	153	L.22:	185	addp
122	instr336	154	instr468	186	instr471

187	indiri	213	instr515	239	.align 4
188	asgni	214	argp	240	_rows:
189	instr468	215	instr509	241	.space 32
190	instr470	216	calli	242	.globl _down
191	indiri	217	addSP(8)	243	.align 4
192	instr471	218	pop	244	_down:
193	indiri	219	L.38:	245	.space 60
194	instr243	220	instr133	246	.globl _up
195	L.17:	221	instr180	247	.align 4
196	L.14:	222	instr259U(L.37)	248	_up:
197	instr133	223	addrgp(L.42)	249	.space 60
198	instr180	224	argp	250	.dataEnd
199	instr259U(L.13)	225	instr509	251	.data
200	L.12:	226	calli	252	.align 1
201	cnsti(99)	227	addSP(4)	253	L.42:
202	ret	228	pop	254	,byte 10
203	.globl _print	229	L.36:	255	.byte 0
204	_print:	230	cnsti(99)	256	.align 1
205	addSP (-4)	231	ret	257	L.41:
206	instr114	232	textEnd	258	.byte 37
207	L.37:	233	.data	259	.byte 99
208	instr122	234	.globl _x	260	.byte 32
209	instr124	235	.align 4	261	.byte 0
210	lshi	236	_x:	262	.dataEnd
211	instr514	237	.space 32		
212	argi	238	.globl _rows		

【図1】

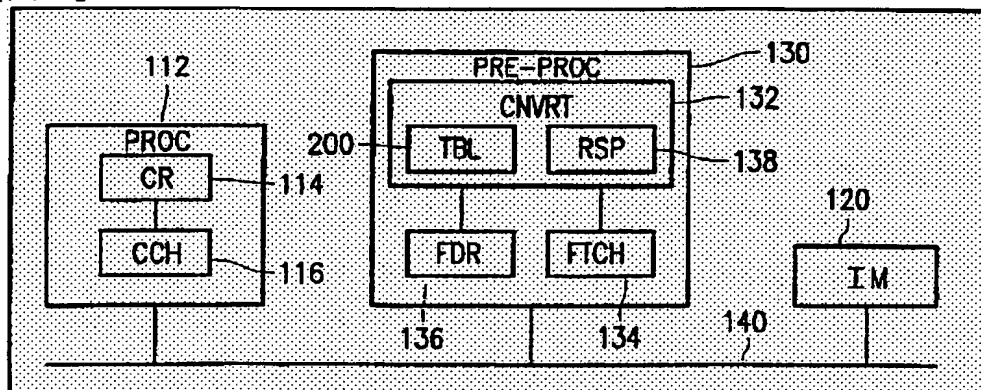


FIG. 1A

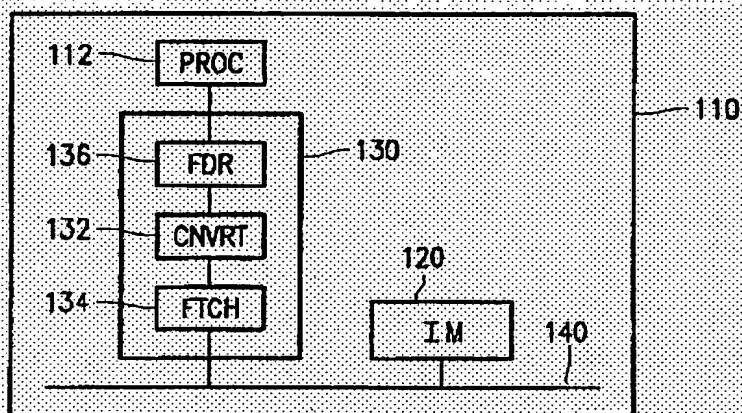


FIG. 1B

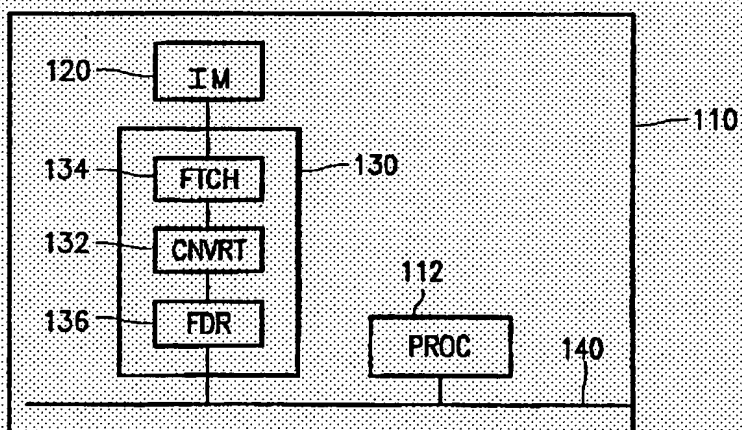
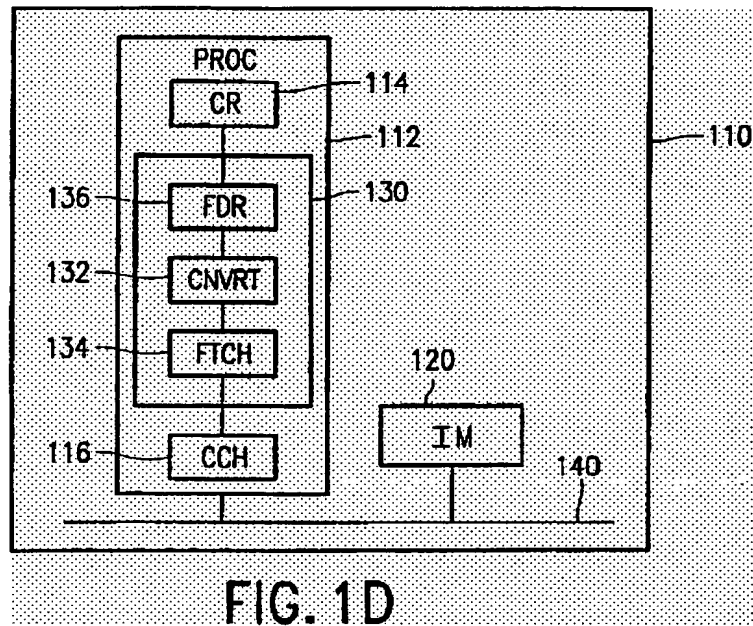
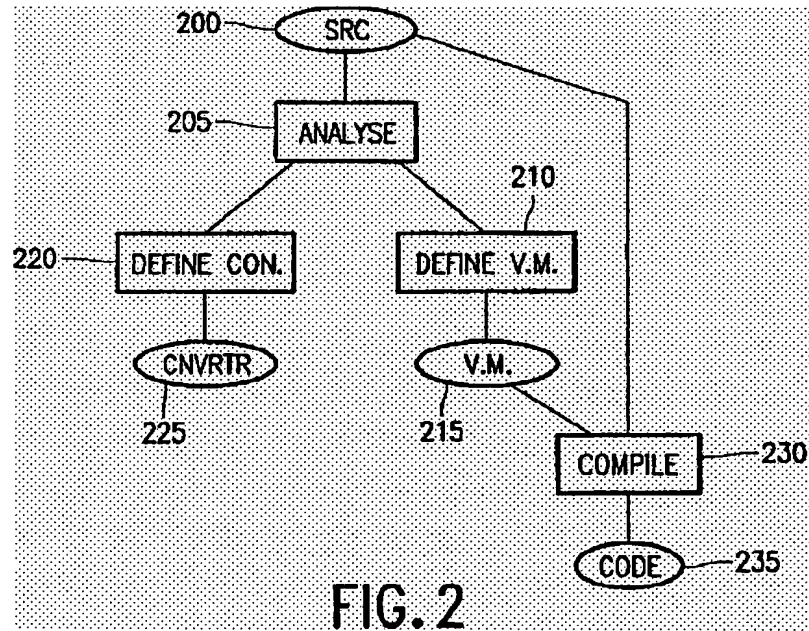


FIG. 1C

【図1】



【図2】



【図3】

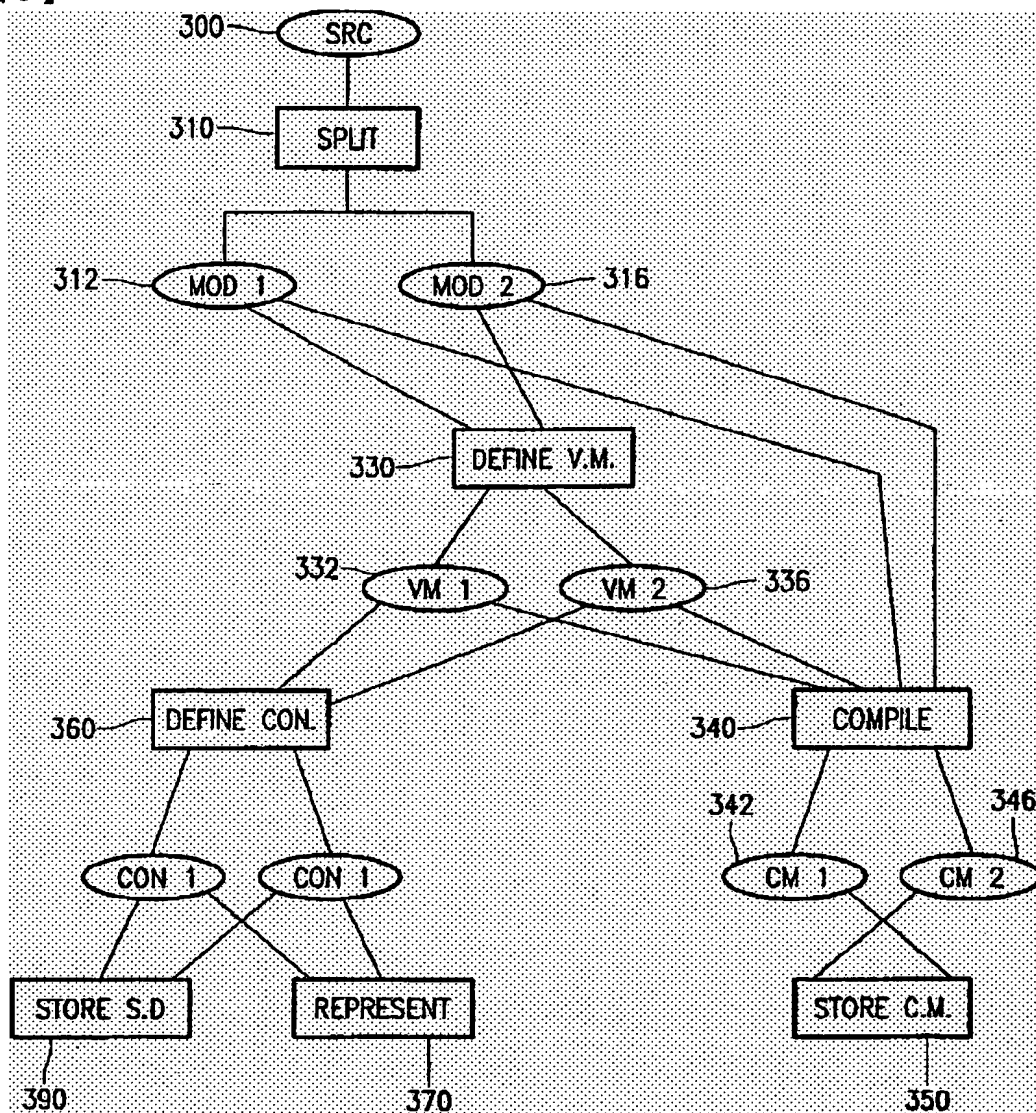


FIG. 3

【図4】

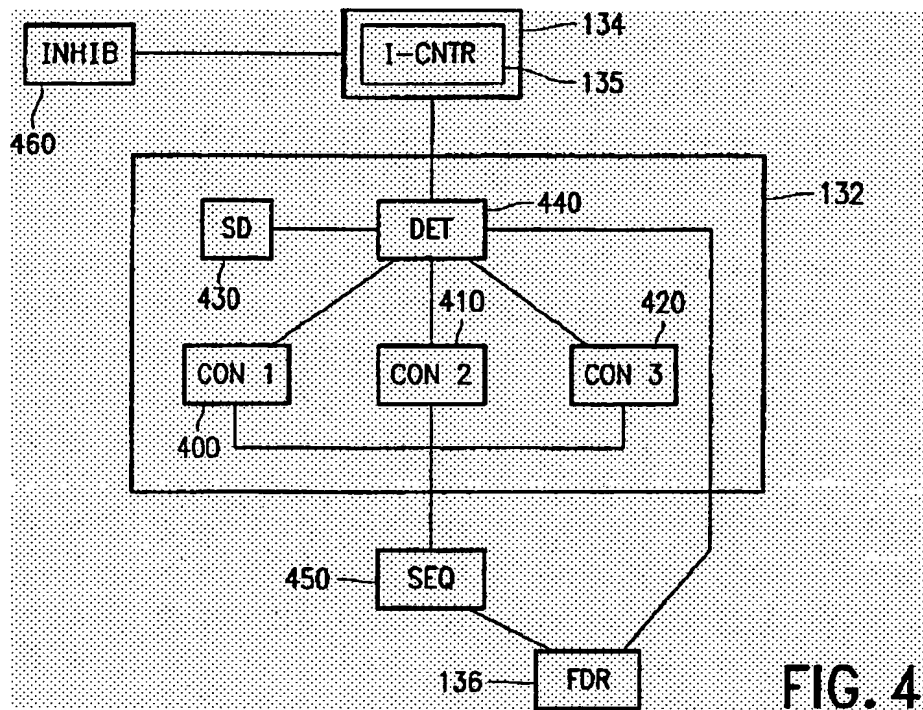


FIG. 4

【図5】

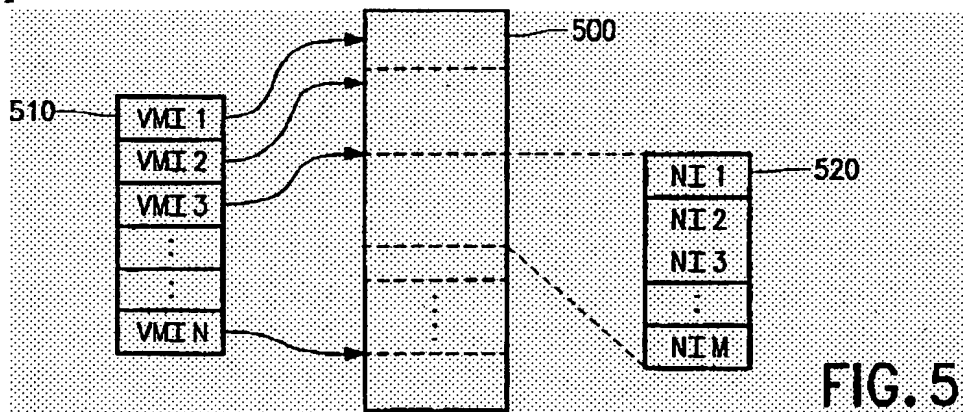


FIG. 5

【国際調査報告】

INTERNATIONAL SEARCH REPORT		International application No. PCT/JP 98/01453
A. CLASSIFICATION OF SUBJECT MATTER		
IPC6: G06F 9/31B, G06F 12/08 // G06F 17/30 According to International Patent Classification (IPC) or to both national classification and IPC		
B. FIELDS SEARCHED		
Minimum documentation searched (classification system followed by classification symbols)		
IPC6: G06F		
Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched		
SE, DK, FI, NO classes as above		
Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)		
DIALOG, PAJ		
C. DOCUMENTS CONSIDERED TO BE RELEVANT		
Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	WO 9727537 A2 (SUN MICROSYSTEMS, INC.), 31 July 1997 (31.07.97), page 4 - page 5, figure 6A, claim 1, 6, 12-13, 19, 23	1-2, 4-6
A	---	3, 7-12
Y	WO 9723823 A2 (PHILIPS ELECTRONICS N.V.), 3 July 1997 (03.07.97), see whole document	1-2, 4-6
A	---	3, 7-12
A	US 5586323 A (SHINOBU KOIZUMI ET AL), 17 December 1996 (17.12.96), column 4, line 1 - column 6, line 65, figures 3, 51, claims 1-3, abstract	1-12
<input checked="" type="checkbox"/> Further documents are listed in the continuation of this C. <input checked="" type="checkbox"/> See patent family sources.		
* Special categories of cited documents: "A" document defining the general state of the art which is not considered to be of particular relevance "B" other document not published on or after the international filing date "L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another claim or other special reason (as specified) "O" document referring to an oral disclosure, use, exhibition or other means "P" document published prior to the international filing date but later than the priority date claimed "I" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention "X" document of particular relevance the claimed invention cannot be considered novel, or cannot be considered to involve an inventive step when the document is taken alone "Y" document of particular relevance the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art "Z" document member of the same patent family		
Date of the actual completion of the international search		Date of mailing of the international search report
21 April 1999		22-04-1999
Name and mailing address of the ISA/ Swedish Patent Office Box 5055, S-102 42 STOCKHOLM Facsimile No. +46 8 666 02 86		Authorized officer Linus Wretblad Telephone No. +46 8 782 23 00

Form PCT/ISA/210 (second sheet) (July 1993)

INTERNATIONAL SEARCH REPORT		International application No. PCT/IB 98/01453
C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT		
Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	Department of Electrical Engineering and Computer Sciences University of California, Berkeley, 94720 The convergence of telecommunications and computing: What are the implications today? David G. Messerschmitt IEEE Proceedings, August 1996 see pages 17-22 ---	1-12
A	NO 9727536 A1 (SUN MICROSYSTEMS, INC.), 31 July 1997 (31.07.97), page 3 - page 4, claims 1, 8-10, abstract -----	1-12

Form PCT/ISA/210 (continuation of second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT				International application No.	
Information on patent family members				PCT/IB 98/01453	
Patent document cited in search report		Publication date	Patent family member(s)		Publication date
WO	9727537 A2	31/07/97	WO	9727536 A	31/07/97
			WO	9727539 A	31/07/97
			WO	9727544 A	31/07/97
WO	9723823 A2	03/07/97	EP	0811190 A	10/12/97
			GB	9526129 D	00/00/00
			US	5872978 A	16/02/99
US	5586323 A	17/12/96	EP	0510616 A	28/10/92
			EP	0834803 A	08/04/98
			EP	0838754 A	29/04/98
			JP	4322329 A	12/11/92
WO	9727536 A1	31/07/97	WO	9727537 A	31/07/97
			WO	9727539 A	31/07/97
			WO	9727544 A	31/07/97

Form PCT/ISA/210 (patent family annex) (July 1992)

フロントページの続き

(81)指定国 EP(AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), JP

(72)発明者 ホリッセン ポーラス エム ハー エム
アー

オランダ国 5656 アーアー アイन्दー
フェン プロフ ホルストラーン 6

(72)発明者 メレンブルークス フランシスカス イエ
ー ハー エム

オランダ国 5656 アーアー アイन्दー
フェン プロフ ホルストラーン 6

(72)発明者 ストラフェルス ポール

オランダ国 5656 アーアー アイन्दー
フェン プロフ ホルストラーン 6

(72)発明者 トレスヘル ヨーアヒム アー

オランダ国 5656 アーアー アイन्दー
フェン プロフ ホルストラーン 6

【要約の続き】

令を、コア (114) により実行されるネイティブ命令に変換するために使用される。